UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingenieria Informática

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA SEGURO DE CONTROL DOMÓTICO

Autor: Laura Cerdá Muñoz
Tutor: Óscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Diciembre 2018

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA SEGURO DE CONTROL DOMÓTICO

Autor: Laura Cerdá Muñoz
Tutor: Óscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Diciembre 2018

Resumen

Resumen

Este proyecto consiste en desarrollar un sistema domótico con seguridad SSL. El objetivo es que cualquier usuario que tenga instalado en su casa este sistema pueda controlar los dispositivos de su hogar de forma remota, lo que quiere decir que conocerá el estado de sus elementos domóticos periódicamente y podrá interactuar con ellos (con aquellos que lo permitan, como los actuadores) enviando órdenes de apagado o encendido desde una web.

Por un lado, se implementará un Gateway que con un servidor recibirá peticiones y mostrará a través de una aplicación Web los datos recogidos. Por otro lado, se ha implementado una centralita a la que se conectan dispositivos que simulan ser los electrodomésticos de una casa particular. Esta última denominada Sistema de Control de Dispositivos Domóticos (SCDD) y es la que aquí se documenta.

Para el sistema Hardware se han seleccionado una Raspberry Pi3 como controlador central y dos dispositivos, un sensor DHT11, que representa todos aquellos dispositivos cuya función es recoger información del entorno doméstico, y un actuador, Relé, que simula todos aquellos elementos que permiten la recepción de órdenes por parte del usuario. Para el sistema software se han implementado varios ficheros en lenguajes PHP y Python.

Los ficheros de codificación del sistema los podemos clasificar en dos grupos según su finalidad. Por un lado los dedicados al servidor web (index.php, register.php, info.php y end.php), implementados en PHP. Y por otro lado los dedicados a lectura de los ficheros de configuración (WiFi_Checker.py) creados previamente con los datos del formulario, los dedicados al almacenamiento de la información recogida de, los dispositivos domóticos (CrearBBDD.py), los dedicados al establecimiento y mantenimiento de la comunicación con el servidor (conectar_dispositivos.py, conectar_sensor.py, conectar_actuador.py, Dispositivo.py) y por último el fichero que implementa el servidor que recibe las órdenes para gestionar el comportamiento de los actuadores (server.py). El flujo de ejecución de este código es el siguiente. En primer lugar se ejecuta el servidor web (con el sistema funcionando como punto de acceso), recoge los datos introducidos en el formulario por el usuario y los almacena en un fichero de configuración llamado *WiFiConnect.cfg*, a continuación se ejecuta el proceso de gestión de la conexión WiFi que configura las credenciales de usuario, en el momento que el usuario finalice la configuración, el sistema automáticamente (desde tareas de cron) se reinicia y comprueba que el sistema tiene conexión a Internet. Por último se ejecuta un script que realiza tres instrucciones, la primera para crear la base de datos con dos tablas Device (para el registro de los dispositivos) y *Activity_Device* (para el registro de la información recogida por los dispositivos), otra llamada que ejecuta el servidor que recoge las órdenes del usuario para interactuar con los actuadores, y por último la ejecución del cliente que realiza las peticiones necesarias al servidor para establecer la comunicación.

Este sistema ofrece comunicación segura gracias a en primer lugar el código de verificación propio del sistema y en segundo lugar a certificados de seguridad SSL tanto de cliente como de servidor.

Palabras Clave

Domótica, relé, sensor, actuador, dispositivo, internet de las cosas, servidor, hardware, software, automatización, android, framework

Abstract

In this project we have developed a domotic system with SSL security.

The goal is for any user who owns this system in their home to remotely control their home devices, which means that they will periodically know the status of their domotic elements and will be able to interact with them.

On the one hand, a Gateway is implemented that receives requests with a server and shows the collected data through a Web application. On the other hand, a software system is implemented on a Raspberry Pi to which devices that simulate home appliances are connected. The latter is called the Domotic Device Control System (SCDD) and is the one documented here.

For the Hardware system, a Raspberry Pi3 has been selected as the central controller and two devices, a DHT11 sensor, which represents all those devices whose function is to collect information from the domestic environment, and an actuator, Relay, that simulates all those elements that allow reception of orders by the user. For the software system, several files have been implemented in PHP and Python languages.

The coding files of the system can be classified into two groups according to their purpose. On the one hand those dedicated to the web server (index.php, register.php, info.php and end.php), implemented in PHP.

And on the other hand those dedicated to reading the configuration files (WiFi_Checker.py) previously created with the data of the form, those dedicated to the storage of the information collected from, the domotic devices (CrearBBDD.py), those dedicated to the establishment and maintenance of communication with the server (connect_devices.py, connect_sensor.py, connect_actuator.py, Device.py) and finally the file that implements the server that receives the commands to manage the behavior of the actuators (server.py).

The execution flow of this code is as follows. First, the web server runs (with the system functioning as an access point), collects the data entered in the form by the user and stores it in a configuration file called *WiFiConnect.cfg*, then executes the WiFi connection management process that configures the user's credentials; at the moment the user finishes the configuration, the system automatically restarts (from the cron tasks) and verifies that the system has an Internet connection. Finally, it executes a script that performs three instructions, the first to create the database with two Device tables (for the registration of the devices) and *Activity_Device* (for the record of the information collected by the devices), another call executed by the server that collects the user's commands to interact with the actuators, and finally the execution of the client that makes the necessary requests to the server to establish communication.

This system offers secure communication thanks to the first verification code of the system and secondly to SSL security certificates for both client and server.

Key words

Domotics, relay, sensor, actuator, device, internet of things, server, hardware, software, automation, android, framework

Agradecimientos

Quiero agradecer en primer lugar a toda mi familia por apoyarme durante todos los años dedicados a esta carrera, en especial a mi hermana Sofía por ofrecerme su ayuda y conocimientos siempre que lo he necesitado.

También quiero le agradezco a mi compañero Diego Chicote por todas las prácticas que hemos llevado a cabo juntos como un equipo durante estos años.

Finalmente agradezco a mi tutor Óscar Delgado por ofrecerme la oportunidad de realizar este proyecto.

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

1.1. Motivación del proyecto

A día de hoy existen varios sistemas domóticos muy similares al que queremos crear [1], sin embargo, la mayoría de estos sistemas carecen de sistemas de seguridad o son muy pobres. No se le está dando demasiada importancia a este tema y debemos ser conscientes de que en un sistema domótico se registran datos muy sensibles y sería un problema serio si caen en manos de ciberdelincuentes.

Estas son las principales razones que llevan a crear un sistema domótico *Open Source* que proporcione seguridad en las comunicaciones entre el usuario y el sistema.

La principal idea es crear un sistema económico cuyo código fuente sea público para que todos los usuarios que lo deseen puedan mejorar y adaptar este sistema domótico en base a sus necesidades.

1.2. Objetivos y enfoque

Los objetivos son conseguir un sistema domótico seguro, de bajo coste, open source para que cualquier persona sin necesidad de conocimientos previos pueda instalarlo en su casa o en caso de que si posea conocimientos, pueda incluso mejorarlo a partir del software realizado en este proyecto.

Este sistema final tendrá la capacidad de crecer y adaptarse a las necesidades de los nuevos tiempos, mejorar la calidad de vida proporcionando comodidad y tranquilidad para los usuarios utilizando certificaciones de seguridad. No sería aceptable ningún tipo de ataque que pudiera poner el riesgo los datos del usuario o en control de su hogar.

1.3. Metodología y plan de trabajo

Los pasos que se van a seguir para la realización del proyecto, serán los especificados a continuación:

En primer lugar debemos tener claro el proyecto que vamos a realizar. El producto final será un sistema domótico que ejecute con los dispositivos las tareas que el usuario indique. Es decir, controlar sus electrodomésticos y elementos domóticos de forma remota.

- Estudio del funcionamiento de los recursos hardware (RPi, sensor, actuador).
- Creación del circuito electrónico que conectará los dispositivos con la centralita (sensores y actuadores con la placa principal) e implementación de scripts bash para el control de los mismos dentro del ámbito local.
- Implementación del cliente HTTP que envíe peticiones de conexión al servidor
- Implementación del software necesario para incorporar la funcionalidad del manejador de dispositivos al cliente http que enviará la información recogida al servidor para que sea visualizada por el usuario desde la aplicación web.
- Implementación de un servidor web que registre los datos necesarios para establecer una conexión del sistema a internet mediante la red WiFi del usuario.

1.4. Organización de la memoria

Esta memoria consta de cuatro capítulos con el orden siguiente:

Estado del arte

En este capítulo se exponen los conocimientos de la actualidad, contemporánea a esté informe sobre temas que engloban al proyecto realizado y aquí documentado. Se pretende ofrecer una visión general sobre este tema para poner en contexto al lector

Sistema, diseño y desarrollo

Aquí se mostrarán los pasos seguidos para el análisis del problema y el diseño y desarrollo de nuestra solución. Dando una explicación detallada de que problemas se han tenido y porque se han tomado según que decisiones.

Se especificará paso a paso cual ha sido el orden de las tareas llevadas a cabo durante el desarrollo del proyecto. Mostrando de forma explicativa la implementación del código. Este apartado permitirá al lector recrear un sistema domótico de estas características sin necesidad de grandes conocimientos previos sobre el tema.

Pruebas y resultados

En este capítulo se expondrán las pruebas realizadas durante el desarrollo, y se mostrarán evidencias del correcto funcionamiento del sistema.

Conclusiones y trabajo futuro

Finalmente analizaremos posibles mejoras y ofreceremos algunas ideas de aplicaciones futuras para este proyecto.

2
Estado del arte

2.1. Introducción

Antes de entrar en materia, debemos comprender el término principal que define este proyecto, **domótica**.

La definición oficial en la Real Academia Española para este término [2]

> *Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda*

se resume en dos palabras, *casa automática*, ya que resulta de la unión de *domus*, 'casa' en latín, y *automática*, 'por sí solo' en griego. Por lo que en definitiva cuando hablamos de domótica nos referimos a una vivienda dotada de un conjunto de sistemas automáticos conectados entre sí que realizan tareas.

Podríamos situar los inicios de la domótica en los primeros sistemas de control remoto, sin embargo, el imparable avance de la tecnología ha hecho que en pocos años estos sistemas evolucionen hasta hacer de nuestra vivienda un hogar inteligente que interactúa con nosotros y nos proporciona seguridad, comodidad y por supuesto ahorro de energía [3].

La domótica es el resultado más conocido actualmente como ejemplo de IoT.

2.2. Historia, nacimiento y evolución del IoT

Podríamos pensar que el Internet de las Cosas es algo moderno y actual, sin embargo, ya en los años 20' **Nikola Tesla** (1856 - 1943), uno de los padres del las comunicaciones inalámbricas, mencionaba el crecimiento de la conectividad global definiendo al mundo como *gran cerebro.*

> "*Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmi... y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo"*

Posteriormente Alan Turing (1912 - 1954) expresó su creencia de que existiría la necesidad de proporcionar inteligencia y capacidades de comunicación a los elementos que nos rodean [4].

> *"...también se puede sostener que es mejor proporcionar la máquina con los mejores órganos sensores que el dinero pueda comprar, y después enseñarla a entender y hablar inglés. Este proceso seguirá el proceso normal de aprendizaje de un niño"*

Sin embargo, el alto coste del hardware y la inmadurez tecnológica de los tiempos que corrían provocaron un gran descenso en estas investigaciones, que se dieron por teorías imposibles de materializar.

Algo más de una década después, de forma tímida, comenzaron a crearse los primeros protocolos de comunicación. Se crearon redes independientes y muy dispares, hasta que en los años 70' **Vinton Gray Cerf** (1943 - actualidad) y **Robert Elliot Kahn** (1938 - actualidad) crearon el protocolo standard TCP/IP [5] que permitió la comunicación global. Al principio su utilización era estrictamente para maniobras militares, pero posteriormente se abrió al mundo.

Hacia finales de los 80', **John Romkey** (fechas desconocidas) creó el primer objeto capaz de recibir ordenes a través de Internet. Se trataba de una tostadora que un usuario podía apagar y encender de forma remota.

Algunos años más tarde en 1999 se escuchó por primera vez en boca de **Kevin Ashton** (1968 - actualidad) la expresión *'Internet of things'* (IoT o Internet de las cosas) que en pocas palabras define, aún a día de hoy, todo el potencial de estas investigaciones y sus posibles aplicaciones, que han cambiado seguirán cambiando nuestra forma de vivir [6, 7, 8].

No fue hasta la globalización de las comunicaciones inalámbricas (principios del siglo XXI) que se comenzó a investigar en profundidad la infinidad de posibilidades que ofrecía esta red llamada Internet.

2.3. Aplicaciones IoT

Hoy en día conocemos aplicaciones del IoT como pueden ser

Sistemas de envíos de paquetes. En este sector existe un claro ejemplo de aplicación IoT. Se realizan seguimientos del los paquetes enviados, permitiendo conocer la localización de cada vehículo de transporte en todo momento. Esto permite, por ejemplo, prevenir los robos de vehículos así como a ojos del comprador, ofrece una información de la localización de su pedido en tiempo real.

Agricultura y ganadería. En este ámbito también se esta introduciendo la tecnología IoT, para realizar seguimientos del sembrado en agricultura, midiendo la temperatura, la humedad, la luminosidad y muchos de los factores que afectan a la siembra. Estos datos facilitan a los agricultores el saber que zona es mejor para la siembra o que producto esta listo para recoger. En ganadería se realizan estudios biométricos de los animales y su geolocalización, esto aporta un gran avance para los ganaderos para conocer la salud de sus animales respecto a su tamaño.

Salud. Gracias al IoT también en medicina se realizan estudios para el control del paciente, sus constantes. Se están desarrollando sistemas de diagnóstico anticipado. La empresa española Libelium ha desarrollado un dispositivo, My Signals, que monitoriza a los pacientes a distancia proporcionando una mejora muy considerable en atención a pacientes de áreas con escasez de médicos.

Domótica. La domótica es uno de los ejemplos de aplicación del IoT que más potencial tiene, sin embargo, aún se encuentran en desarrollo debido a las dificultades por incompatibilidad entre elementos de distintas marcas o por infraestructura. No obstante, poco a poco esta tecnología va creciendo por buena dirección y van surgiendo nuevos productos. [9].

2.4. Vulnerabilidades del IoT

Desde el boom del IoT no han pasado demasiados años, sin embargo, ya conocemos múltiples casos de ataques a los elementos conectados. Con cada nuevo objeto conectado se crea también un nuevo objetivo para los ciberdelincuentes o coloquialmente llamados *hackers*.

Los datos son preocupantes, se han dado casos de poca envergadura como, secuestros de dispositivos conectados mediante bluetooth (ej. altavoces, auriculares. etc.) o incluso ataques más importantes y dañinos como robo de datos, mediante el acceso de intrusos a los asistentes domóticos de un edificio en los cuales se almacena gran cantidad de información. [10].

A continuación se comentan algunos ejemplos de casos reales y experimentos que demuestran las vulnerabilidades del IoT.

Ataque a los sistemas de ventilación

El ataque se produjo a principios de 2012 en una fábrica del gobierno de New Jersey. Los ciberdelincuentes consiguieron hacerse con el control remoto de los termostatos del edificio y modificaron la temperatura. Por suerte no hubo daños humanos ni materiales, sin embargo, podría haber sido un suceso mucho más grave de haber sido por ejemplo en un centro de proceso de datos [11].

Ataque DDoS masivo contra DynDNS

El ataque a DynDNS, una empresa estadounidense proveedora de direcciones DNS para direcciones IP dinámicas, fue provocado el 21 de Octubre del año 2016 en Estados Unidos. En este ataque los ciberdelincuentes infectaron dispositivos de IoT como cámaras IP o impresoras con un malware que provocó un ataque por **Denegació de servicios** (DDoS) que ocasionó la caída de múltiples servicios como, Amazon, PayPal, Netflix y un largo etcétera. [12, 13, 14, 15, 16]

El malware utilizado fue un troyano denominado Mirai enfocado al IoT que utiliza una botnet para realizar ataques DDoS, realiza escaneos reiterados en toda la red con el objetivo de encontrar los dispositivos más vulnerables [17].

Control remoto de un avión

En 2013 se realizó un experimento para demostrar la facilidad de ocasionar graves daños con solo un dispositivo android. El autor del experimento fue **Hugo Teso** (1984 - actualidad), consultor de seguridad informática Y piloto comercial, que mediante una aplicación android desarrollada sobre un framework capaz de insertar información en los planes de vuelo, consiguió crear un simulador de vuelo. Utilizando la información del servicio Flightradar24 [18] que monitoriza la posición de vuelos comerciales, la aplicación podía interferir lo sistemas de cualquier avión dentro de un cierto rango [19, 20, 17].

Estos son solo tres ejemplos de las vulnerabilidades del IoT, pero actualmente se habla de que el 70 % de los dispositivos de IoT utilizados más comúnmente contienen vulnerabilidades significativas.

2.5. Soluciones IoT

Por desgracia, a día de hoy aún no existen propuestas de soluciones para superar las vulnerabilidades del IoT, debido a que las distintas marcas de dispositivos no se ponen de acuerdo para establecer un protocolo estándar de comunicación segura.

Se valoran soluciones como mecanismos de autorización y autenticación mediante contraseñas seguras y perecederas, o de almacenamiento de datos limitando la cantidad de información registrada en los dispositivos o controles por desbordamiento de buffer, fuzzing o denegación de servicio [21].

3

Sistema, diseño y desarrollo

3.1. Análisis y diseño

3.1.1. Exposición del problema

La idea de este proyecto es crear un sistema domótico que permita a un usuario interactuar con los elementos domóticos de su casa, a través de cualquier dispositivo con acceso a internet.

Este proyecto se divide en dos partes, el lado del servidor, situado en el Gateway, y el lado del cliente, que gestiona los dispositivos, que llamaremos SCDD (**S**istema de **C**ontrol de **D**spositivos **D**omóticos).

Para esta parte del proyecto, SCDD, necesitamos elaborar un sistema que controle y registre los datos de los elementos domóticos, ya sean *actuadores*, cualquier tipo de objeto que reciba órdenes de programación, encendido o apagado, o *sensores*, de temperatura, de movimiento o cualquier tipo de objeto que simplemente recoja información del entorno en el que se encuentra.

Además debemos implementar un proceso que permita conectar este sistema a la red ya que necesita comunicarse con un servidor. Sabiendo que carece de teclado, monitor y cualquier forma de ver el contenido de la máquina.

3.1.2. Propuesta de la solución

Para proponer una solución primero dividiremos el trabajo en tres partes, por un lado resolveremos la manera de comunicar la centralita (RPi) con los elementos. Por otro lado, la forma en la cuál el usuario conecta el sistema domótico a su red WiFi y por último implementaremos la comunicación con el servidor.

Para el desarrollo utilizaremos una placa Raspberry Pi 3, y para simular los dispositivos del edificio utilizaremos un sensor de temperatura DHT11 y un Relé como actuador.

Conexión a la placa principal

Para la primera parte debemos documentarnos sobre el funcionamiento de los pines gpio de la placa y la distribución de las conexiones de los dispositivos y posteriormente aplicar lo aprendido mediante scripts independientes que controlen dichos elementos.

Conexión del sistema a la red WiFi

En segundo lugar, resolveremos la conexión a la nuestra red wifi. La idea es crear un pequeño servidor web que pida al usuario las credenciales de su red. Sin embargo, para que el usuario pueda acceder a este servidor, el propio sistema tiene que tener un acceso a una red, no seria necesario que tenga internet. Para resolver este problema el propio sistema crearía un punto de acceso, es decir, generaria su propia red wifi. De esta forma el usuario, conociendo las credenciales de esta red, podría conectarse y acceder al servidor. Una vez configurado el wifi en el sistema, este anularia su punto de acceso y se conectaría a la red del hogar.

Comunicación con el servidor

Con los dos puntos anteriores resueltos, ya solo nos quedaría implementar un cliente que que comunique con el servidor. Este cliente deberá realizar una primera petición de conexión al servidor y posteriormente realizar peticiones periódicas que indiquen el estado de los dispositivos, de esta forma el usuario final tendría cada poco tiempo información actualizados. Sin embargo, aquí no solo se debe implementar un cliente, recordemos que el sistema también dispone de actuadores por lo que debe permitir la recepción de peticiones para el control sobre estos.

3.1.3. Diagramas

Previo al desarrollo del sistema se han creado dos diagramas donde se especifica el diseño de este sistema Figura 3.1 y Figura 3.2.

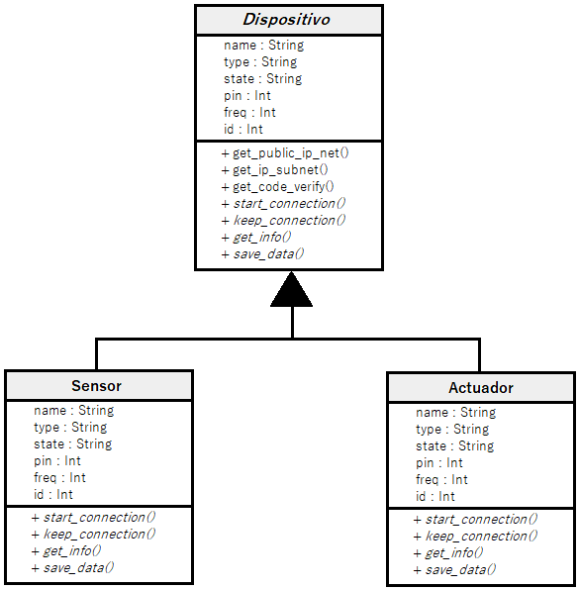

Figura 3.1: Diagrama de clases para crear los dispositivos domóticos

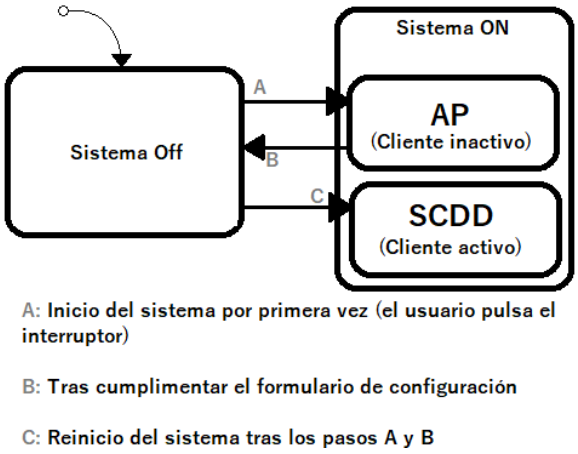

Figura 3.2: Diagrama de estado para el SCDD

Procesador	GPU	Memoria	Conectividad inalámbrica	Conectividad de red	Puertos
Broadcom BCM2837 Cortex-A53 (ARMv8) 64-bit SoC	VideoCore IV 400 MHz	1GB LPDDR2 SDRAM	2.4GHz IEEE 802.11.b/g/n Bluetooth 4.1	Fast Ethernet 10/100 Gbps	GPIO 40 pines HDMI 4 x USB 2.0 CSI (cámara Raspberry Pi) DSI (pantalla tácil) Toma auriculares / vídeo compuesto Micro SD Micro USB (alimentación)

Tabla 3.1: Características técnicas de la placa Raspberry Pi 3B

3.2. Sistema Hardware

3.2.1. Placa principal: Raspberry Pi 3 (RPi3)

Los orígenes de la Raspberry Pi se encuentran en el Reino Unido, en la Universidad de Cambridge en 2011, pero no comenzó a comercializarse hasta 2012. Se trata de un pequeño ordenador de bajo coste y bajo consumo cuyo objetivo era fomentar la enseñanza de la informática [22].

A nivel de software, este mini ordenador permite la ejecución de sistemas operativos basados en Linux sin olvidar las posibilidades que ofrece a nivel de hardware incluyendo 40 pines GPIO. Estas características la hacen idónea para su utilización en sistemas robóticos.

El la Figura 3.3 se muestra el modelo de placa utilizada para este desarrollo. se trata de una *Raspberry Pi 3B* cuyas características se muestran a continuación Tabla 3.1 [23].

3.2.2. Sensor: DHT11

Utilizamos como ejemplo de elemento de tipo sensor un sensor de temperatura y humedad como el de la Figura 3.4 que tiene las siguientes características Tabla 3.2.

3.2.3. Actuador: Relé

Utilizamos como ejemplo de elemento de tipo actuador un Relé como el de la Figura 3.5 que tiene las siguientes características mostradas en la Tabla 3.3.

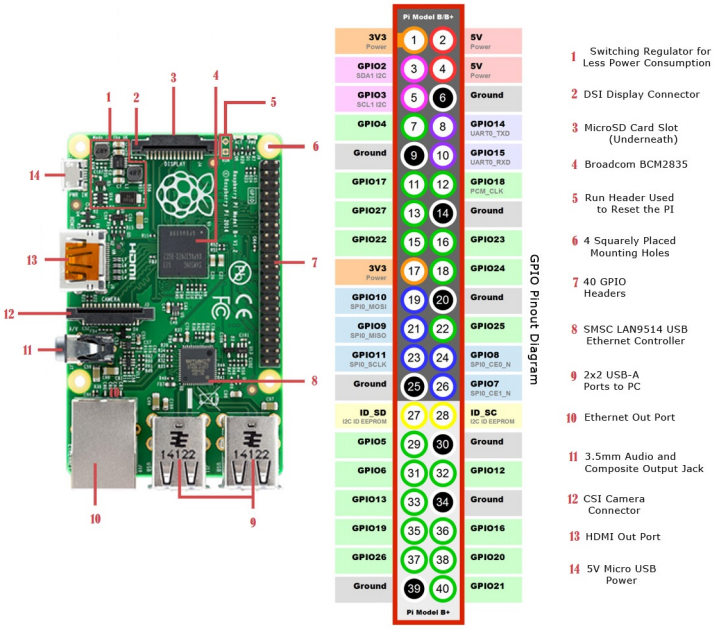

Figura 3.3: Distribución de los pines GPIO

Alimentación	3Vdc $\leq$ Vcc $\leq$ 5Vdc
Rango de medición de temperatura	0 a 50 ^{o}C
Precisión de medición de temperatura	$\pm$2.0 ^{o}C
Resolución Temperatura	0.1^{o}C
Rango de medición de humedad	20
Precisión de medición de humedad	4
Resolución Humedad	1

Tabla 3.2: Características técnicas del sensor DHT11

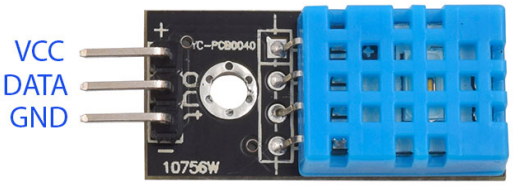

Figura 3.4: Sensor de temperatura y humedad, modelo DHT11

Voltaje	5V
Canales	2
Indicadores	LED

Tabla 3.3: Características técnicas del Relé

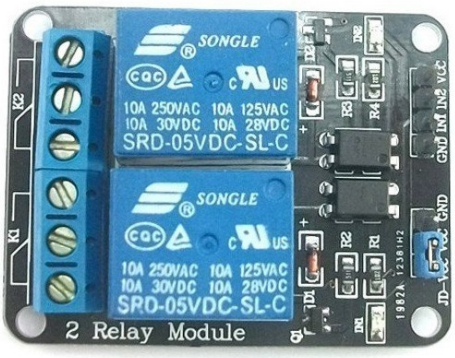

Figura 3.5: Actuador, Relé

3.2.4. Montaje

Una vez conocemos las características de los componentes que necesitamos, realizamos el circuito que conectará la placa con los dispositivos para posteriormente controlarlos y conseguir información de ellos mediante un pequeño desarrollo software.

En primer lugar conectamos el sensor de temperatura Figura 3.6 como se explica a continuación. Con un cable de color negro conectaremos el pin GND del sensor al pin 14 (Ground) de la RPi, el pin Vcc del sensor al pin 1 (3V3 Power) de la RPi y finalmente conectaremos el pin de datos, es decir, el pin DATA del sensor a un pin de transmisión de datos del la RPi, en este caso hemos elegido el pin 16 (GPIO 23).

Para terminar el circuito montaremos las conexiones entre el Relé y la placa Figura 3.6. Conectaremos el pin GND del relé al pin 6 (Ground) de la RPi, a continuación conectaremos el pin Vcc al pin 2 (5V) de la RPi de nuevo debemos conectar el pin para la transmisión de datos, entre el relé y la RPi en este caso se ha utilizado el pin 11 (GPIO 17) de la RPi. Cabe señalar que aunque el relé tiene un indicador led, también tiene la posibilidad de conectar una bombilla o un elemento externo sobre el que actuar, en este caso se ha utilizado un diodo led que simulará una bombilla.

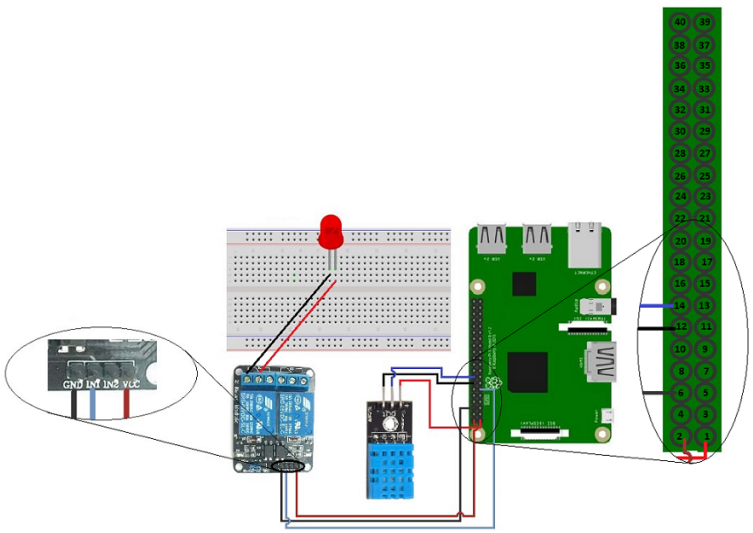

Figura 3.6: Montaje del circuito para la conexión entre centralita y dispositivos

3.3. Desarrollo Software

3.3.1. Software inicial de la placa RPi3

Cualquier placa RPi viene de fábrica sin ningún tipo de software. Para iniciarla primero debemos instalarle sus sistema operativo Raspbian OS gratuito y basado en una distribución de GNU/Linux. [24].

Raspbian es el sistema operativo recomendado y pensado para estas placas, ya que viene con más de 35,000 paquetes [25], software precompilado en un formato de fácil la instalación. Se trata de una distribución ligera para moverse ágilmente en el hardware de la Raspberry pi [26].

Algunos de los paquetes preinstalados útiles para este proyecto son:

- Interprete de lenguaje Python
- Paquetes PHP
- Librerías de apache2 como servidor web
- Navegado web
- Editor de texto

3.3.2. Base de datos

Para el almacenamiento de la información recogida por los diferentes dispositivos se emplea una base de datos SQLite3 ya que es una biblioteca que trabaja directamente con ficheros y no es necesario ningún proceso que funcione como servidor. Ofrece un buen rendimiento realizando las transacciones más rápido que, por ejemplo, **MySql** o **PostgreSQL** [27], permite trabajar desde múltiples lenguajes entre ellos Python y además es de dominio público.

La base de datos creada para el sistema contiene toda la información necesaria para la interacción con los dispositivos, para ello se han creado dos tablas *device* Tabla 3.4 y *activity_device* Tabla 3.5.

Id	Almacena el identificador del dispositivo asignado por el servidor
Pin	Registra el numero del GPIO de la RPi al cual esta conectado el dispositivo
Name	Registra el nombre del dispositivo, formado por la unión del *Type* y el *Pin*
Type	Indica el tipo de dispositivo con un carácter, Sensor 'S' o Actuador 'A'

Tabla 3.4: Descripción de los campos de la tabla *device*

Id	Almacena el identificador del dispositivo asignado por el servidor
Ddate	Registra la fecha en la cual se ha leído la actividad del dispositivo
Ttime	Registra la hora en la cual se ha leído la actividad del dispositivo
Info	Almacena los datos recogidos por o del dispositivo

Tabla 3.5: Descripción de los campos de la tabla *activity_device*

3.3.3. Lenguajes y Librerías

Lenguajes

Para el desarrollo de este sistema se han utilizado dos lenguajes de programación.

Python - Se ha seleccionado este lenguaje para la implementación del código correspondiente al funcionamiento general del sistema, es decir, la comunicación con el servidor, y la gestión de los dispositivos. Se ha elegido Python ya que posee unas características idóneas para el desarrollo de este proyecto, principalmente porque es un lenguaje orientado a objetos sencillo pero con mucho potencial, por ejemplo para la gestión de ficheros de configuración y posee múltiples librerías.

HTML/PHP - Se ha utilizado para el desarrollo del servidor Web inicial, al cual accede el usuario para configurar las credenciales necesarias para el sistema.

Librerías

Las librerías de Python utilizadas son las siguientes:

- *requests.* Utilizada para la implementación del cliente.
- *RPi.GPIO.* Utilizada para la gestión de los GPIO de la RPi.
- *Adafruit.* Utilizada para la lectura de los datos del sensor de temperatura y humedad DHT11
- *http.server.* Utilizada para la implementación del servidor dedicado a recoger las peticiones de el usuario sobre los actuadores.
- *simplejson.* Utilizada para gestionar los mensajes JSON de las peticiones.
- *ConfigParser.* Utilizada para parsear el fichero de configuración donde se establecen las credenciales de la red WiFi del usuario.
- *urllib2.* Utilizado para interacción con webs, para averiguar la IP pública
- *os.* Para acceder a funcionalidades dependientes del Sistema Operativo, como acceder a un fichero.
- *sys.* Utilizada para acceder a funcionalidades relacionadas directamente con el intérprete
- *subprocess.* Para trabajar de forma directa con del sistema operativo
- *time.* Utilizada para calcular el el periodo de comunicación entre una comunicación y la siguiente.
- *sqlite3.* Utilizada para realizar consultas sobre la base de datos.
- *ABCMeta.* Utilizada para realizar métodos abstractos.

Es un lenguaje de código abierto y muy estricto en sus sintaxis, lo que proporciona orden y legibilidad a nuestro código.

3.3.4. Codificación

En esta sección se explicará la implementación de cada una de las partes del SCDD. Dividiremos esta sección en tres puntos, que corresponden con los pasos que se han seguido para este desarrollo [28]:

- Configuración del Punto de Acceso o AP
- Scripts Python de configuración y auxiliares
- Conexión del sistema a la red WiFi del usuario
- Comunicación con el servidor

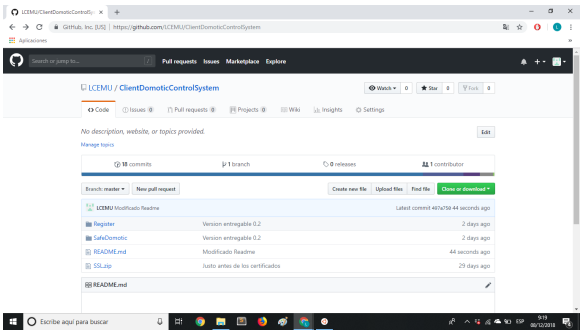

Figura 3.7: Proyecto en GitHub, ilustra aquí la localización del código para el servidor Web y el código para el cliente como sistema SCDD

Configuración del Punto de Acceso o AP

Primero de todo se configura la placa como un punto de acceso, para ello debemos modificar ciertos archivos del sistema e instalarnos dos programas *hostapd* y *dnsmasq* con el siguiente comando [29].

```
sudo apt-get install hostapd
sudo apt-get install dnsmasq
```

A continuación seguiremos los siguientes pasos:

A. Nos aseguramos de que dichos programas no estén ejecutando:

```
sudo systemctl stop hostapd
sudo systemctl stop dnsmasq
```

B. Configuramos una IP estática editando el fichero /etc/dhcpcd.conf y escribiendo las siguientes líneas Figura 3.8. En nuestro caso hemos definido la ip 192.168.4.1.

```
sudo nano /etc/dhcpcd.conf
```

```
# IP ESTATICA (CODIGO QUE ACTIVA EL Access Point)
interface wlan0
static ip_address=192.168.4.1/24
nohook wpa_supplicant
```

Figura 3.8: Líneas añadidas al fichero *dhcpcd.conf*

C. Substituimos el contenido del fichero de configuración dnsmasq.conf por defecto por las siguientes lineas. De esta forma definiremos el rango de direcciones a asignar por el punto de acceso que como podemos ver irán desde la 192.168.4.2 a la 192.168.4.20.

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

```
interface=wlan0      # Use the require wireless interface - usually wlan0
  dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
```

Figura 3.9: Líneas añadidas al fichero *dnsmasq.conf*

D. A continuación editaremos el fichero /etc/hostapd/hostapd.conf con el siguiente comando, para definir las características de nuestra red y sus credenciales Figura 3.10.

```
sudo nano /etc/hostapd/hostapd.conf
```

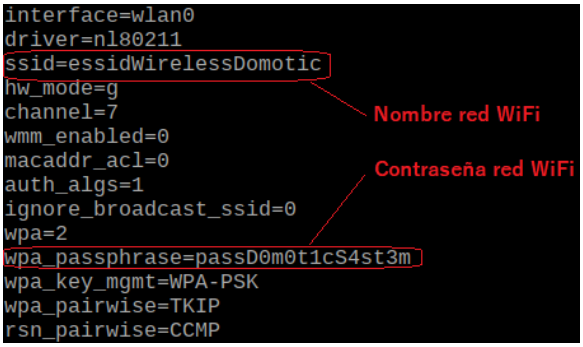

Figura 3.10: Líneas añadidas al fichero *hostapd.conf*

E. Y para que e sistema conozca la ruta del fichero de configuración de la red editamos el fichero /etc/default/hostapd con el siguiente comando y añadimos la variable DAEMON_CONF con el valor de dicha ruta Figura 3.11

```
sudo nano /etc/default/hostapd
```

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Figura 3.11: Líneas añadidas al fichero *hostapd*

F. En caso de que conectemos la placa a través de un cable Ethernet debemos redirigir todo el tráfico que llegue a nuestra wlan0, por lo que de nuevo debemos modificar otro fichero de configuración y descomentar la siguiente línea Figura 3.12.

```
sudo nano /etc/sysctl.conf
```

G. Agregamos enmascaramiento de IP para eth0 con tablas IP y editamos el fichero /etc/rc.local para que se carguen dichas tablas en cada reinicio del sistema.

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo sh -c ïptables-save > /etc/iptables.ipv4.nat"
```

```
net.ipv4.ip_forward=1
```

Figura 3.12: Líneas añadidas al fichero *sysctl.conf*

Scripts python de configuración y auxiliares

CrearBBDD.py - Este script python crea la base de datos *DomoticaDevice.db* y ejecuta las instrucciones sqlite correspondientes a la creación de las tablas *device* Figura 3.13, que almacena los dispositivos existentes en el sistema y *activity_device* Figura 3.14 que almacena la información que recogen los dispositivos. Este script se ejecutará automáticamente, como explicaremos más adelante.

```
CREATE TABLE device
    (ID INT PRIMARY KEY NOT NULL,
    PIN TEXT NOT NULL,
    NAME TEXT NOT NULL,
    TYPE CHAR(1) NOT NULL);
```

Figura 3.13: Código SQL para la creación de la tabla que registra los dispositivos

```
CREATE TABLE activity_device
    (ID INT NOT NULL,
    DDATE TEXT NOT NULL,
    TIME TEXT NOT NULL,
    INFO CHAR(50) NOT NULL);
```

Figura 3.14: Código SQL para la creación de la tabla que registra la actividad de los dispositivos

Constantes.py - Es el módulo donde se establecen las constantes del sistema, como los nombres de los ficheros de configuración, la IP donde se localiza el servidor, el formato de la información que se envía al servidor etc.

Wifi_Checker.py - Este script se encarga de realizar las modificaciones necesarias para establecer las credenciales introducidas por el usuario en la configuración de la conexión WiFi del sistema SCDD, modificando el fichero */etc/Network/interfaces* con la función *save_credentials(name, psw)* Tabla 3.6. A su vez desactivará el SCDD como punto de acceso (AP o Access Point). Este fichero es ejecutado por el servidor web en el momento que se confirman los datos introducidos.

Connect_Checker.py - Este script se ejecutará en cada reinicio del sistema SCDD con una tarea programada en *cron* con la siguiente instrucción.

```
@reboot python /path/SafeDomotic/Connect_Checker.py
```

Con él, comprobamos si el SCDD está conectado correctamente al WiFi del usuario, es decir, implícitamente comprueba si las credenciales introducidas por el usuario son correctas en caso contrario, volverá a activar el SDCC como punto de acceso y lo reiniciará. Para realizar esta operación se han implementado dos funciones Tabla 3.7.

ControladorAP.py y ***ConfigAP.py*** - Estos script se encargan de gestionar la activación y desactivación de la AP como punto de acceso. *ControladorAP.py* recibirá un parámetro, cuyos posibles valores son on/off y respecto al valor recibido ejecutará la activación o desactivación llamando a las funciones *get_estado_AP()*, *onAP()* y *offAP()* de *ConfigAP.py* Tabla 3.8.

Función	Descripción
save_credentials(name, psw)	Modifica el fichero */etc/network/interfaces*, introduciendo las credenciales de WiFi del usuario a continuación de las líneas `wpa-ssid` y `wpa-psk`.

Tabla 3.6: Funciones implementadas en Wifi_Checker.py

Función	Descripción
comprobarConexion()	que se encargará de consultar si el sistema tiene una IP asignada, lo que implica una conexión a un router. Si esta función devuelve un valor correcto el programa continúa. Si por otro lado, la función devuelve error se invocará desde el main la función *delete_credentials()*.
delete_credentials()	Eliminará las credenciales del WiFi del fichero */etc/network/interfaces*.

Tabla 3.7: Funciones que implementadas en Connect_checker.py

Función	Descripción
get_estado_AP()	Esta función se encarga de comprobar cual es el estado del fichero */etc/dhcpcd.conf*. Es decir, si el sistema SCDD esta configurado como AP o no.
onAP()	Modifica el fichero */etc/dhcpcd.conf* incluyendo las líneas que activan el AP.
offAP()	Modifica el fichero */etc/dhcpcd.conf* eliminando las líneas que activan el AP.

Tabla 3.8: Funciones implementadas en implementadas en ConfigAP.py

Conexión del sistema a la red WiFi del usuario

Para la funcionalidad que debe conectar el SCDD a la red WiFi del usuario se ha realizado un servidor web que permite al usuario configurar dicha conexión en cuatro pasos. Se han implementado por lo tanto cuatro ficheros en lenguaje PHP especificados a continuación.

Index.php - Fichero de bienvenida al sistema e información del proceso que debe seguir el usuario para completar esta configuración con éxito.

Register.php - Fichero que muestra un formulario con los tres campos, que se corresponden con *Nombre de la red WiFi*, *Password de la red WiFi* y *Código de verificación*. Los dos primeros campos se rellenarán con las credenciales del WiFi como describe el nombre, y en el tercer campo se introducirá un código único que proporciona autenticación entre el SCDD y el Gateway.

Info.php - Mostrará una tabla de resumen de las credenciales introducidas para que el usuario las confirme. En caso de que exista algún error el usuario tendrá la posibilidad de corregirlo mediante un botón que le retornará a la pantalla anterior para que pueda corregirlo antes de enviar el formulario

End.php - Es el fichero que ejecutará el registro, es decir, confirmará las credenciales y ejecutará un script Python que realizará las modificaciones de configuración necesarias para que el SCDD se conecte a la red WiFi del usuario y reiniciará el sistema.

Comunicación con el servidor

Esta parte del código se ejecuta automáticamente tras la configuración de la conexión WiFi, del mismo modo que *Connect_checker.py*. Para ello incluimos otra tarea en *cron* como la siguiente:

```
@reboot sh /path/SafeDomotic/domotico.sh
```

domotico.sh - Este fichero es un script bash que ejecutará tres instrucciones. En primer lugar creará la base de datos

```
python ./BBDD/CreaBBDD.py
```

a continuación ejecutará el servidor de recepción de órdenes On/Off en background, para que continúe ejecutándose el script bash sin esperar a que este servidor se detenga.

```
python ./server/server.py &
```

y por último ejecutará el cliente

```
python ./conectarDispositivos.py
```

conectar_actuador.py y ***conectar_sensor.py*** - Estos son dos pequeños script que contienen un programa principal o *main* donde crean una instancia de la clase Actuador o Sensor correspondientemente. Estos programas inicializan cada objeto indicando los parámetros, *name*, *commands*, que indicarán si el dispositivo espera peticiones del usuario, *estado inicial*, *frecuencia*, que será el tiempo que transcurra entre una comunicación y la siguiente, y el *pin*, el GPIO del cual el sistema tendrá que leer la información del dispositivo.

conectar_dispositivos.py - Este script se encarga de crear las líneas de ejecución que llamarán a *conectar_actuador.py* y *conectar_sensor.py* para crear todos los dispositivos a nivel de programa. Es decir, se puede entender como el programa que comienza el proceso de registro de nuevos dispositivos en la base de datos y establecimiento de la comunicación del SCDD con el Gateway. Tras crear las líneas de ejecución se realizará la propia ejecución, con subprocesos mediante la iteración del array creado.

Dispositivo.py - Es el fichero que contiene las definiciones de los dispositivos, tanto sensores como actuadores. En este script se utilizan objetos, es decir, se define una clase Dispositivo abstracta y dos clases que la implementan, Sensor y Actuador.

Class Dispositivo. Se trata de una clase abstracta que implementa las funciones genéricas Tabla 3.9 que utilizarán todos los dispositivos y además define funciones abstractas que deberán implementar cada uno de los dispositivos con sus propias características.

Funciones comunes

Función	Descripción
obtener_ip	obtiene la ip pública y local del sistema SCDD
get_code_verify	Consigue el código de verificación que el cliente ha introducido en el formulario previamente y en este momento esta almacenado como parte de la configuración del sistema

Tabla 3.9: Funciones implementadas en *Dispositivo.py*

Funciones específicas para cada dispositivo

Función	Descripción
iniciar_conexion()	Envío del comando inicial, POST, para establecer conexión con el servidor. Construye un mensaje JSON Figura 3.15 con la información inicial del dispositivo y la configuración que se va a establecer, como el periodo de tiempo entre comunicaciones, *freq* o los comandos permitidos *commands*, así como el código de verificación en la cabecera del mensaje. Como respuesta del servidor espera un identificador único que será asignado a dicho dispositivo.
mantener_conexion()	Una vez abierta la comunicación con el servidor, el cliente realizará peticiones periódicas para mantener la conexión abierta. En este caso enviará comandos PUT con mensajes JSON Figura 3.16 que contengan la información actualizada de los dispositivos en cada momento y de nuevo el código de verificación en la cabecera del mensaje.
obtener_informacion()	Se encarga de capturar la información de los sensores y actuadores que están en funcionamiento que más tarde se enviará al servidor.
guardar_datos()	La información que se le envía a servidor se almacena en una base de datos local, esta función se encarga de actualizar las tablas correspondientes.

Tabla 3.10: Funciones de *Dispositivo.py*, definidas como abstractas en la clase Dispositivo e implementadas en las clases Actuador y Sensor

Class Sensor. Se trata de una clase que implementa a la clase dispositivo con las características

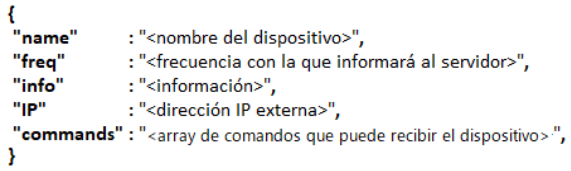

Figura 3.15: Estructura del mensaje JSON para el inicio de la comunicación

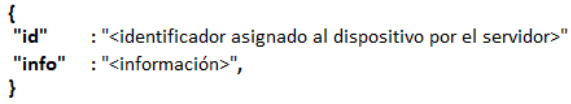

Figura 3.16: Estructura del mensaje JSON para el mantenimiento de la comunicación

específicas de un sensor Tabla 3.10.

Class Actuador. Implementa los mismos métodos que la clase sensor pero específicos para dispositivos de tipo actuador Tabla 3.10.

server.py - Implementa un servidor que espera peticiones POST con un mensaje JSON. La estructura del JSON esta muy definida y debe contener el **id** del dispositivo cuyo estado se quiere modificar y la *acción* (On/Off). Con esta información la placa RPi ejecutará las instrucciones necesarias para la activación o desactivación del GPIO correspondiente utilizando las siguientes instrucciones siguientes en orden [30].

1. Establecimiento del modo de numeración de los GPIO, en este caso BCM:
`GPIO.setmode(GPIO.BCM)`

2. Indicamos de que forma vamos a utilizar el GPIO, en modod de entrada, *IN*, o en modo de salida *OUT*. Modo de salida en este caso.
`GPIO.setup(<numGPIO>, GPIO.OUT)`

3. Con las dos instrucciones anteriores el GPIO quedará configurado, pero para activarlo o desactuvarlo debemos utilizar las siguientes instrucciones.

`GPIO.output(<numGPIO>, GPIO.HIGH)` (Activación)
`GPIO.output(<numGPIO>, GPIO.LOW)` (Desactivación)

Como hemos comentado al comienzo de este informe gran parte de la motivación para realizar este proyecto ha sido la aplicación de IoT con controles de seguridad, por eso hemos desarrollado un sistema con seguridad a través de certificados SSL[REFERENCIA CONCEPTOS DE LOS SSL] ya que proporcionan confidencialidad. Además hemos incluido un código de verificación para autenticar al usuario en el momento de la conexión del sistema a su red inalámbrica.

La utilidad del código de verificación es la de autenticar al SCDD como parte complementaria correspondiente al Gateway, es decir, el sistema completo lo componen el Gateway, que almacena el servidor, y el SCDD y sólo deben comunicarse entre ellos. Por lo tanto el código de verificación debe ser único y si es erróneo porque un intruso quiera acceder de manera ilícita, la comunicación no se establecerá.

Desde el punto de vista del usuario se utilizará de la siguiente manera: el Gateway llevará una identificación física en una etiqueta con un código *CV* Figura 3.17, esta identificación sera única y aleatoria para cada par (Gateway-SCDD), el usuario deberá introducir dicho código en el formulario comentado

anteriormente, de este modo quedará configurado el sistema junto con las credenciales de red inalámbrica.

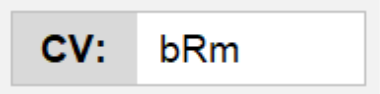

Figura 3.17: Etiqueta localizada en el Gateway o Etiqueta EG, que identifica al par Gateway-SCDD

3.3.5. Software inicial del sistema domótico

- Software inicial de raspbian
- Apache2
- Paquetes PHP
- Librerías utilizadas para la implementación del SCDD.
- Funcionalidad de control de entorno domótico

De esta forma hemos creado a partir de una placa con funcionalidad básica Debian, en un elemento que controla los dispositivos del hogar a elección del usuario.

3.4. Flujo de ejecución

En la siguiente Figura 3.18, se ilustra el proceso de ejecución del sistema SCDD.

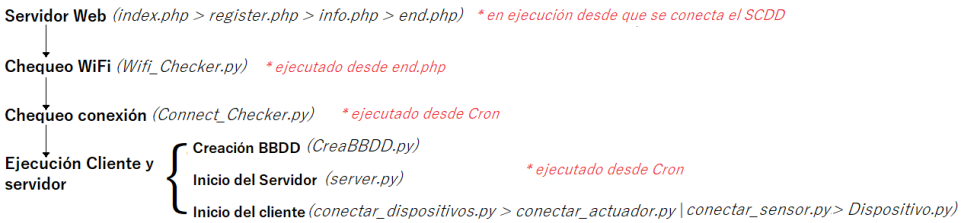

Figura 3.18: Esquema de ejecución de ficheros del SCDD

3.5. Organización de directorios del sistema

A continuación observamos la estructura de los ficheros que componen el SCDD Figura 3.19 Figura 3.20

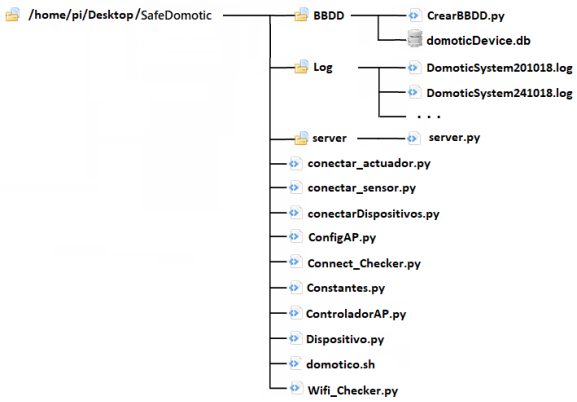

Figura 3.19: Estructura de los directorios del proyecto, Cliente

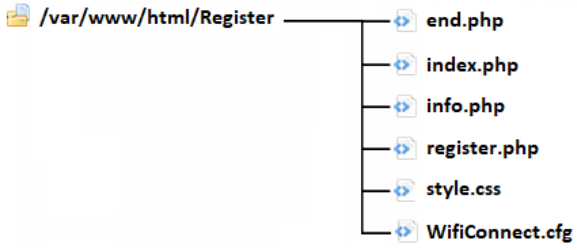

Figura 3.20: Estructura de los directorios del proyecto, Servidor Web

4

Pruebas y Resultados

Para comprobar si el sistema creado funciona correctamente se han realizado en primer lugar pruebas con cada parte del proyecto, es decir, por un lado se comprueba el proceso por el cual es usuario realiza la configuración del sistema a través del servidor web y por otro lado la comunicación entre cliente y servidor incluyendo la recepción de peticiones on/off por parte del SCDD.

4.1. Configuración del sistema por el usuario

Para realizar estas pruebas primero debemos conocer el proceso que el usuario final debe seguir para configurar su sistema:

- Instalación del sistema.
- Conexión a la red WiFi generada por el SCDD.
- Configuración del SCDD.

Instalación del sistema

El usuario obtiene el sistema compuesto de un Gateway y un SCDD, debe conectarlo a la red eléctrica de las dependencias donde quiera utilizarlo Figura 4.1. El sistema permite que el Gateway y el SCDD estén tanto en las mismas dependencias como en distintas, es decir, en la misma subred o en subredes distintas, no obstante para la segunda opción el usuario deberá abrir dos puertos uno en cada router de cada dependencia.

Conexión a la red WiFi generada por el SCDD

Con el sistema conectado a la red eléctrica, el usuario deberá encender el SCDD pulsando el interruptor que lleva incorporado. A continuación, deberá conectar un dispositivo con acceso a WiFi a la red Figura 4.3 que generará el SCDD cuyas credenciales se encuentran en la etiqueta ES, Figura 4.2, donde ESSID es el nombre de la red WiFi y PASS es su contraseña.

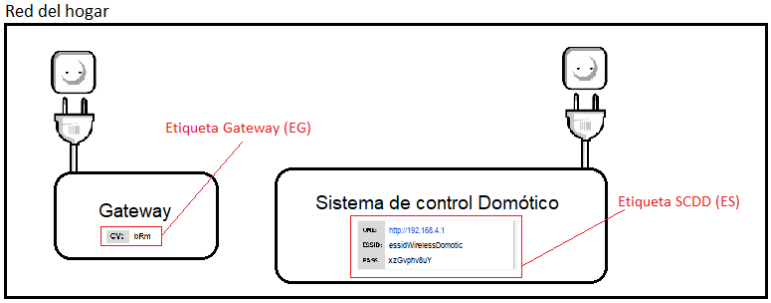

Figura 4.1: Paso 1 - Instalación del sistema

URL:	http://192.168.4.1/domotica/index.php
ESSID:	essidWirelessDomotica
PASS:	passD0m0t1cS4st3m

Figura 4.2: Etiqueta situada en el SCDD o Etiqueta ES

A nivel de programa, los datos introducidos por el usuario son recogidos por un PHP y almacenados en un fichero de configuración *WifiConnect.cnf* Figura 4.4 que a su vez ejecuta un programa Python *WiFi_Checker.py*. Este fichero realizará los cambios necesarios para desactivar el punto de acceso (AP) generado por el propio sistema, modificando el fichero */etc/dhcpcd.conf* Figura 3.8 y registrando las credenciales leídas del fichero de configuración *WifiConnect.cnf* en */etc/network/interfaces* Figura 4.5. Una vez registrados estos datos el sistema se reiniciará automáticamente para hacer efectiva la conexión a la red del usuario. De este modo el SCDD ya estará preparado para establecer comunicación con el Gateway.

Configuración del SCDD

Ya estamos conectados al AP del SCDD, y ahora el usuario debe configurar la conexión de este a su propia red WiFi. Para ello debemos acceder a la URL indicada en la etiqueta ES y completar todos los pasos que se indican. Se deberán introducir las credenciales de nuestra red WiFi y un código de verificación (CV) indicado en la etiqueta EG, situada en el Gateway.

4.2. Comunicación SCDD-Gateway

Tras completar todos lo pasos anteriores, el usuario ya dispondrá de un sistema domótico funcional, es decir, el sistema automáticamente comenzará a enviar peticiones de comunicación al Gateway y de este modo el usuario ya podrá monitorizar los elementos domóticos de su hogar u oficina.

A nivel de programa, tras realizar todos los pasos previos de configuración y conexión a la red el sistema ejecutará un primer script python *Connect_Checker.py* que comprobará si el sistema está conectado correctamente a la red y por tanto tiene acceso a internet, en caso contrario volverá a reiniciar el SCDD para que el usuario pueda corregir los errores que existan en los datos introducidos. En caso de que sea correcto se ejecutará el cliente, también implementado en Python que realizará peticiones al servidor. Para el establecimiento de la conexión el cliente enviará peticiones de tipo POST, con un JSON como el de la Figura 3.15 y periódicamente seguirá enviando este tipo de peticiones hasta que el servidor acepte la comunicación. Una vez aceptada la comunicación, el cliente comenzará a enviar peticiones PUT con un JSON como el de la Figura 3.16

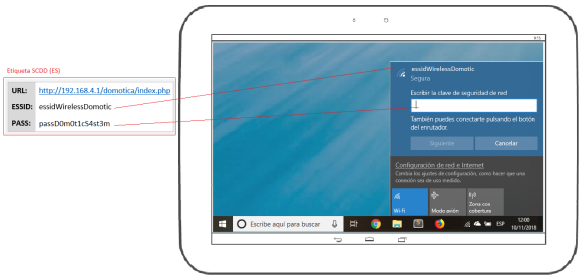

Figura 4.3: Paso 2 - Conexión a la red WiFi generada por el SCDD

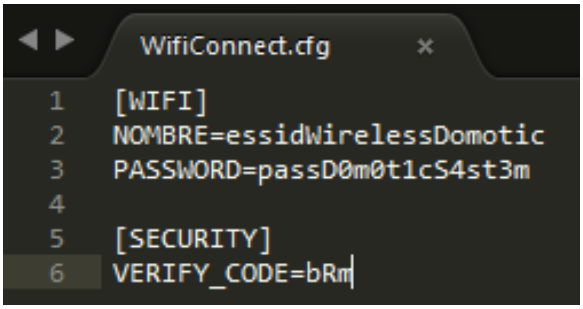

Figura 4.4: Fichero de configuración creado con las credenciales del WiFi del usuario y el código de verificación del sistema

4.3. Pruebas de integración

Lado del cliente o SCDD

Podemos comprobar en la Figura 4.9 el proceso que sigue el sistema durante su ejecución. Se observan evidencias del estado del SCDD a partir del partir de su configuración.

Líneas 1 a 6 - Partimos de que el usuario ya ha completado el formulario de configuración con las credenciales de su red WiFi y el código de verificación. Ahora el sistema lee el fichero de configuración establece las credenciales en el fichero correspondiente (*/etc/network/interfaces*).

Línea 7 - Tras un reinicio esperado y automático el sistema comprueba si las credenciales introducidas son correctas. Como vemos en esta traza sí son correctas por lo que el sistema continua su flujo, en otro caso volvería a reiniciarse para permitir al usuario volver introducir sus credenciales.

Líneas 8 a 21 - Finalmente se ejecuta el cliente. Comprobamos por las líneas 10 y 11 de la Figura 4.9 que para establecer la comunicación el SCDD envía una petición POST con un JSON Figura 3.15 que define ciertos parámetros, como ya hemos comentado anteriormente el el punto *3.3.4 Codificación* Tabla 3.10. En respuesta a estas peticiones iniciales recibe un identificador para cada dispositivo (líneas 12 y 13). Una vez tiene estos identificadores comienza a enviar peticiones PUT con un JSON Figura 3.16 distinto al inicial que identifican el dispositivo que se esta actualizando y su información. Como vemos estas peticiones se realizan periódicamente, en este caso cada 5 segundos como hemos indicado en el JSON inicial.

Lado del servidor o Gateway

Las Figuras 4.10 y 4.11 muestran desde el lado del servidor como llegan las peticiones desde el sistema SCDD. Como vemos los mensajes se reciben correctamente por lo que el servidor puede utilizarlos para

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
      wpa-ssid essidWirelessDomotic
      wpa-psk passD0m0t1cS4st3m
```

Figura 4.5: Paso 2.1 - Conexión a la red WiFi generada por el SCDD

Figura 4.6: Paso 3.1 - Configuración del SCDD

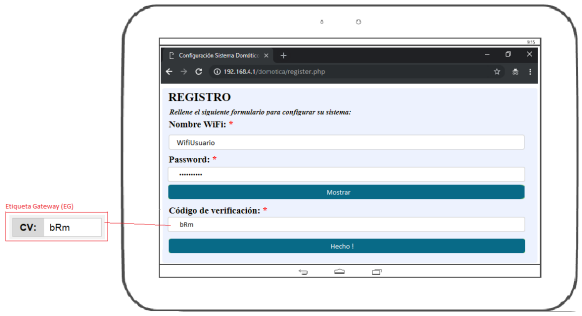

Figura 4.7: Paso 3.2 - Configuración del SCDD

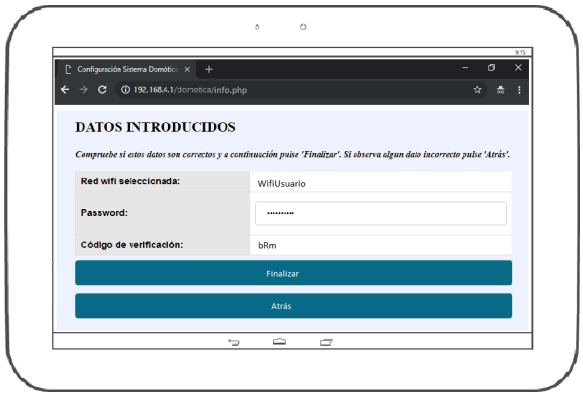

Figura 4.8: Paso 3.3 - Configuración del SCDD

mostrárselos al usuario.

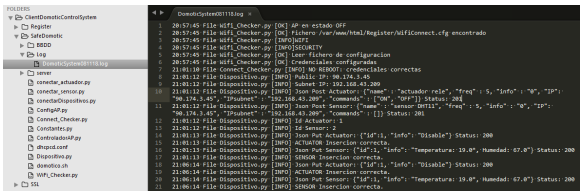

Figura 4.9: Evidencias en un fichero Log de una ejecución completa

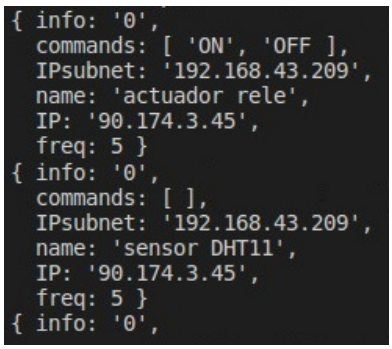

Figura 4.10: Muestra de las peticiones POST de establecimiento de comunicación que llegan al servidor

Bases de datos de SCDD

Comprobamos en la Figura 4.12 que aquellos dispositivos que se han comunicado con el servidor quedan registrados en el sistema SCDD. Estos registros se actualizan al tiempo que los dispositivos recogen la información y la envían al servidor.

```
PUT /brimo/api/devices/1 200 11.117 ms - 8
{ info: 'Temperatura=19.0*, Humedad=67.0', id: 2 }
PUT /brimo/api/devices/2 200 11.026 ms - 8
{ info: 'Disable', id: 1 }
PUT /brimo/api/devices/1 200 371.421 ms - 8
{ info: 'Temperatura=19.0*, Humedad=67.0', id: 2 }
PUT /brimo/api/devices/2 200 11.060 ms - 8
{ info: 'Disable', id: 1 }
```

Figura 4.11: Muestra de las peticiones PUT de mantenimiento de comunicación que llegan al servidor

```
sqlite> SELECT * FROM device;
ID          PIN         NAME        TYPE
----------  ----------  ----------  ----------
1           17          Rele        A
2           23          TempHum     S
```

```
sqlite> SELECT * FROM activity_device;
ID          DDATE       TIME        INFO
----------  ----------  ----------  ------------------
1           08/11/2018  21:01:13    Disable
2           08/11/2018  21:01:13    T: 19.0* - H: 67.0
1           08/11/2018  21:06:14    Disable
2           08/11/2018  21:06:14    T: 19.0* - H: 67.0
```

Figura 4.12: Evidencia de almacenamiento en BBDD de los dispositivos conectados al sistema

5

Conclusiones y trabajo futuro

Este proyecto servirá de base para nuevos proyectos de IoT, hasta ahora tenemos un sistema que realiza acciones según las peticiones de un usuario, por medio de un servidor web al que accede desde un terminal. Sin embargo, existen muchas formas en las que un usuario puede comunicarse con el sistema. Una posible mejora sería adaptar este proyecto para que las peticiones se pudieran realizar mediante un sistema ágil como puede ser un canal de chat, mediante el cuál el usuario podría escribir comandos con un patrón concreto y simplemente enviando el mensaje le llegue a este sistema como una orden. Esto proporcionaría una comodidad extra para el usuario.

Como mejora en medidas de seguridad, se podría fijar un límite máximo de intentos al establecimiento de conexión con el servidor y en caso de que fallen *n* veces por el código de verificación el sistema se bloquea. Para esta actualización, debería de existir algún modo de recuperación del sistema.

Este sistema actualmente permite dos dispositivos, uno de cada tipo, sin embargo, una tarea pendiente sería permitir al usuario que decidiera, dentro de las capacidades de la RPi (número de GPIO), cuantos dispositivos quiere conectar. Esto debería realizarse a nivel de código modificando simplemente el fichero de creación de dispositivos *conectarDispositivos.py*, y recoger por parámetro con un patrón definido cuantos dispositivos se desean conectar con todos sus datos, nombre, tipo, etc

Glosario de acrónimos

- **IoT**: Internet of Things o Internet de las cosas
- **AP**: Punto de acceso
- **WiFi**: Wireless Fidelity
- **RPi**: Raspberry Pi
- **GPIO**: General Purpose Input/Output o Entrada/Salida de Propósito General
- **BCM**: Broadcom, los pines se numeran por la correspondencia en el chip Broadcom (que es la CPU de la Raspberry Pi)
- **SCDD**: Sistema de control de dispositivos domóticos
- **Router**: Dispositivo que proporciona conectividad a nivel de red
- **URL**: Uniform Resorce Locator
- **Android**: Sistema operativo que se emplea en dispositivos móviles
- **Framework**: Entorno de trabajo que proporciona una serie de herramientas
- **JSON**: JavaScript Object Notation o Notación de Objetos de JavaScript
- **Background**: Ejecución en segundo plano.
- **SSL**: Secure Sockets Layer (capa de sockets seguros)
- **DDoS**: Denegación de servicios
- **Malware**: Malicious software, todo tipo de programa o código informático malicioso cuya función es dañar un sistema o causar un mal funcionamiento
- **Botnet**: Grupo de PC's infectados y controlados por un atacante de forma remota

Bibliografía

[1] Carlos Polanco. Televisión secuestrada. *El mundo*, 2018. https://www.elmundo.es/economia/vivienda/2018/03/09/5aa15a8a46163f12448b466c.html.

[2] RAE. *Domótica*. http://dle.rae.es/srv/fetch?id=E7W0v9b.

[3] Domoprac. *Historia de la Domotica*. Domoprac, 2018. http://www.domoprac.com/protocolos-de-comunicacion-y-sistemas-domoticos/historia-de-la-domotica-pasado-presente-y-futuro.html.

[4] Alan Mathison Turing. Computing Machinery and Intelligence. In JSTOR, editor, *Oxford Mind Journal*, chapter Computing. Oxford University Press on behalf of the Mind Association, 1950. http://phil415.pbworks.com/f/TuringComputing.pdf.

[5] Robert Elliot Cerf, Vint and Kahn. TCP/IP. http://history-computer.com/Internet/Maturing/TCPIP.html.

[6] Bruno Cedón. El origen del IoT, 2017. http://www.bcendon.com/el-origen-del-iot/.

[7] María Alejandra Medina C. Historia de la IoT. *El Espectador*, 2017. https://www.elespectador.com/tecnologia/la-historia-detras-de-la-internet-de-las-cosas-articulo-716678.

[8] Postscapes. Cronología IoT. *Postscapes*, 2018. https://www.postscapes.com/internet-of-things-history/.

[9] Luis del Valle Hernández. Las aplicaciones de IoT. https://programarfacil.com/podcast/aplicaciones-del-iot-reales/.

[10] Iván Martín Barbero. Vulnerabilidad IoT. *El PaÍs*, 2016. https://cincodias.elpais.com/cincodias/2016/10/26/lifestyle/1477469985_167878.html.

[11] Gwen Moran. Ataque a una fábria de New Jersey. *Fast Company*, 2013. https://www.fastcompany.com/3008148/cybercriminals-hack-factory.

[12] Nicole Perlroth. Hackers Used New Weapons to Disrupt Major Websites Across U.S. *New York Times*, 2016. https://www.nytimes.com/2016/10/22/business/internet-problems-attack.html.

[13] Nicky Woolf. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian*, 2016. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[14] Antonio M. Martín. Ataque a DynDNS. *El independiente*, 2016. https://www.elindependiente.com/economia/2016/10/25/asi-fue-ataque-informatico-bloqueo-acceso-spotify-netflix-twitter/.

[15] Esther Mucientes. Así se gestó el ciberataque más grave de los últimos 10 años. *El Mundo*, 2016. https://www.elmundo.es/tecnologia/2016/10/22/580b10e5268e3e06158b45e0.html.

[16] Wikipedia. Afectados por ataque a DynDNS, 2016. https://es.wikipedia.org/wiki/Ciberataque_a_Dyn_de_octubre_de_2016#Servicios%5C_afectados.

[17] Mirai (malware). https://www.osi.es/es/servicio-antibotnet/info/mirai/.

[18] Svenska Resenätverket AB. Flightradar24, 2007. https://www.flightradar24.com/60,15/6.

[19] J. J. Velasco. Experimentos de Intrusion en el control de un avión, 2013. https://hipertextual.com/2013/04/como-hackear-un-a.

[20] Expansión. Un experto muestra cómo controlar un avión desde un teléfono con android. *Expansión*, 2013. https://expansion.mx/tecnologia/2013/04/12/un-experto-muestra-como-controlar-un-avion-desde-un-telefono-con-android.

[21] Ruben Ramiro. Ciberseguridad. 2018. https://ciberseguridad.blog/recomendaciones-de-ciberseguridad-en-iot/.

[22] Universidad politécnica de Valencia. Historia de la placa Raspberry Pi, 2013. https://histinf.blogs.upv.es/2013/12/18/raspberry-pi.

[23] Luis Llamas. Características de la placa Respberry Pi. 2017. https://www.luisllamas.es/que-es-raspberry-pi/.

[24] Raspbian. Instalación Raspbian. https://www.raspberrypi.org/documentation/installation/installing-images/.

[25] Debian. Paquetes Raspbian, 2018. https://packages.debian.org/stable/.

[26] Características Raspbian. https://espberry.wordpress.com/2016/03/07/instalar-mis-aplicaciones-en-raspbian/.

[27] Comparación BBDD. https://db-engines.com/en/system/MySQL%3BPostgreSQL%3BSQLite.

[28] Laura Cerdá Muñoz. ClientDomoticControlSystem (Código fuente), 2018. https://github.com/LCEMU/ClientDomoticControlSystem.git.

[29] Stephen Lovely. Creación Access Point. https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/.

[30] Anonymous. Instrucciones GPIO. https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/.

A

Manual de ususario

El sistema que acaba de adquirir esta compuesto por dos partes, Gateway y SCDD (sistema de control de dispositivos domóticos), cada uno de ellos tiene identificación. En Este manual se explican los pasos que se deben seguir para iniciar el SCDD:

1. Conecte el SCDD a la red eléctrica con el cable.
2. Presione el interruptor incorporado a dicho cable.
3. Compruebe que su SCDD dispone de una etiqueta como la siguiente, donde se facilita una URL y unas credenciales (ESSID y PASSWORD).

URL:	http://192.168.4.1/domotica/index.php
ESSID:	essidWirelessDomotica
PASS:	passD0m0t1cS4st3m

4. Utilice cualquier dispositivo con navegador (Explorer, Firefox, Chrome, etc.) y acceso a WiFi y siga en orden los siguiente puntos:
   a) Conecte dicho dispositivo a la red WiFi suyas credenciales están indicadas en la etiqueta del SCDD.
   b) Acceda desde el navegador a la dirección URL indicada en la etiqueta del SCDD.
5. Compruebe que su Gateway dispone de una etiqueta como la siguiente, donde se facilita un código de verificación (CV).

CV:	bRm

6. En la web a la que acaba de acceder deberá cumplimentar un formulario donde se le solicitarán unas credenciales, y un codigo. En los dos primeros campos debe introducir el nombre y la password respectivamente de su propia red WiFi (la de su hogar). Mientras que el el último campo debe introducir el codigo de verificación indicado en la etiqueta del Gateway. Siga las instrucciones de las sucesivas pantallas y pulse 'Finalizar'.

7. Espere aproximadamente 2 min a que el sistema realice el proceso de configuración internamente. Transcurrido este tiempo, el SCDD estará listo para su utilización.

8. Ahora deberá seguir las instrucciones de configuración del Gateway para finalizar la configuración del sistema completo.